Principes et architecture des réseaux

IUT de Villetaneuse — R&T 1re année

Responsable du cours : Laure Petrucci

Polycopié rédigé par : Étienne André, Giulio Manzonetto et Laure Petrucci.

30 novembre 2017

Table des matières

Chapitre 1

Concepts de base

1.1 Généralités

Le terme *informatique* provient d'*information* et d'*automatique*, l'informatique étant le *traitement automatique de l'information.*

Un *réseau* est une organisation de voies de communication entre différentes entités. Cette définition est générale et peut s'appliquer par exemple aux réseaux routiers, ferroviaires, de télécommunications, etc.

Définition 1 (Réseau d'ordinateurs) *Un* réseau d'ordinateurs *est un ensemble d'ordinateurs autonomes interconnectés au moyen d'une seule technologie. Deux ordinateurs sont dits* interconnectés *s'ils peuvent échanger des informations.*

Remarque 1 *Ni l'Internet, ni le World Wide Web (www) ne sont des réseaux d'ordinateurs. L'Internet est plutôt un réseau de réseaux, alors que le World Wide Web est un système distribué qui opère au dessus de l'Internet.*

De manière plus générale, les entités qui communiquent au sein d'un *réseau informatique* sont des ressources informatiques dont on distingue deux types :

- les ressources *matérielles* :
  - *composants de traitement :* ordinateurs, tablettes tactiles, imprimantes, scanners, ...
  - *composants de transmission :* modems, cartes réseaux, commutateurs, routeurs, câbles, ...
- les ressources *logicielles* : applications informatiques, jeux, bases de données, ...

Un réseau informatique est constitué des moyens à la fois matériels et logiciels mis en œuvre pour assurer les communications entre des ressources informatiques.

Un réseau informatique permet aux entités reliées de partager des informations, les résultats de traitements, les ressources, par exemple pour que plusieurs utilisateurs travaillant sur des ordinateurs différents puissent utiliser la même imprimante.

1.2 Représentation de l'information

1.2.1 Quelle information représenter ?

Supposons qu'une machine doive envoyer l'image de la figure 1.1 à une autre, après avoir convenu de la taille de cette image et de l'ordre d'envoi des éléments la constituant. La description se fera, par exemple, carré par carré, ligne par ligne, en commençant en haut à gauche, pour finir en bas à droite. Il est en effet impossible d'envoyer l'image telle quelle sans la coder. La séquence de couleurs à envoyer est donc (en notant blanc B et noir N) :

NNNNN NBBBN NBNBN NBBBN NNNNN

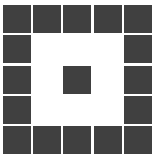


FIGURE 1.1 – Une image à transmettre

Une manière de coder la couleur de chaque carré consiste à associer une valeur à chaque couleur possible, par exemple 1 à B (le pixel sur l'écran est allumé) et 0 à N (le pixel est éteint). La suite de chiffres codant l'image est alors :

00000 01110 01010 01110 00000

1.2.2 Représentation des données

Les *données informatiques* sont représentées par des *suites de nombres*. Ces nombres sont écrits en *binaire* (c'est-à-dire en base 2). En *base* 2, on n'utilise que les chiffres 0 et 1.

L'utilisation de la base 2 garantit de pouvoir représenter un état stable d'un système physique, par exemple :

— circuit électrique ouvert/fermé
— carte perforée avec un trou/sans trou
— ...

Par conséquent, sur un système informatique, les données sont représentées par une suite de chiffres 0 et 1 correspondant à des états différents sur le support physique. Ces états peuvent être des tensions différentes.

1.2.3 Unités utilisées

Définition 2 (bit) *Un* symbole binaire *(donc en base* 2*) est appelé un* bit *(***bi***nary digi***t***).*

1 **bit** permet de coder 2 états : 0 et 1 ;

2 **bits** permettent de coder 4 états : 00, 01, 10 et 11 ;

3 **bits** permettent de coder 8 états : 000, 001, 010, 011, 100, 101, 110 et 111 ;

...

n **bits** permettent de coder 2^n états.

Définition 3 (octet) *Une* suite de 8 bits *est appelée un* octet.

Attention, en anglais le bit est appelé *bit*, alors que l'octet est appelé *byte* !

Les unités multiples des bits et des octets sont décrites dans les tableaux 1.1 et 1.2 :

Unité	Symbole	Valeur (bits)
kilo-bit	Kb	$10^3 = 1\,000$
méga-bit	Mb	$10^6 = 1\,000\,000$
giga-bit	Gb	$10^9 = 1\,000\,000\,000$
téra-bit	Tb	$10^{12} = 1\,000\,000\,000\,000$

TABLE 1.1 – Unités multiples des bits

Unité	Symbole	Valeur (octets)
kibi-octet	Kio	$2^{10} = 1\,024$
mébi-octet	Mio	2^{20}
gibi-octet	Gio	2^{30}
tébi-octet	Tio	2^{40}

TABLE 1.2 – Unités multiples des octets

Traditionnellement, lorsque les préfixes « kilo », « méga », « giga » et « téra » sont appliqués aux octets, ils ne représentent pas une puissance de 10, mais une puissance de 2. Cet usage reste largement en vigueur chez les professionnels comme le grand public. Cependant cette tradition viole les normes en vigueur qui imposent d'utiliser les préfixes « kibi », « mébi », « gibi », « tébi » pour les puissances de 2.

1.3 Mesures de performance

1.3.1 Débit

Définition 4 (débit) *Le* débit *d'un réseau mesure la* quantité d'information *que le réseau peut transmettre* par unité de temps *:*

$$\textit{débit} = \frac{\textit{quantité d'information}}{\textit{temps}}$$

L'unité est par conséquent le *bit par seconde*, noté b/s ou $b.s^{-1}$. Les réseaux actuels ayant un débit assez élevé, on utilise plus souvent des méga-bits par secondes, notés Mb/s ou $Mb.s^{-1}$.

Définition 5 (débits nominal et utile)
- *Le* débit nominal *d'un réseau est la quantité théorique maximale d'information pouvant être transmise par unité de temps.*
- *Le* débit utile *est la quantité d'information effectivement transmise par unité de temps.*

Définition 6 (taux d'utilisation) *Le* taux d'utilisation *du réseau est donc le rapport du débit utile au débit nominal :*

$$\textit{taux d'utilisation} = \frac{\textit{débit utile}}{\textit{débit nominal}}$$

Le taux d'utilisation est inférieur à 100%. Ceci est dû entre autres aux pertes sur la voie de communication et à l'intervalle de temps laissé entre l'envoi de deux messages.

1.3.2 Temps d'acheminement des messages

Définition 7 *Le* temps total *d'acheminement d'un message se compose de deux parties :*
- *le* temps de transmission *est le temps mis pour transmettre la quantité d'information du message, c'est-à-dire :*

$$\textit{temps}_{\textit{transmission}} = \frac{\textit{quantité d'information}}{\textit{débit}}$$

- *le* temps de propagation *est le temps mis pour que le signal se propage sur le matériel. Les équipements traversés peuvent introduire des* retards*.*

$$\textit{temps}_{\textit{propagation}} = \frac{\textit{distance parcourue}}{\textit{vitesse}} + \textit{retards}$$

On a donc :

$$\text{temps}_{\text{total}} = \text{temps}_{\text{transmission}} + \text{temps}_{\text{propagation}}$$

1.4 Classification des réseaux

Les réseaux sont caractérisés non seulement par leur débit, mais également par le *rayon de couverture géographique* qu'ils permettent d'atteindre. Les différentes caractéristiques sont présentées dans le tableau 1.3.

Sigle	Nom	Distance	Débit
PAN	Personal Area Network	quelques mètres	$1Mb/s$
LAN	Local Area Network	jusqu'à $2km$	de $10Mb/s$ à $1Gb/s$
MAN	Metropolitan Area Network	jusqu'à $100km$	environ $100Mb/s$
WAN	Wide Area Network	milliers de km	quelques Mb/s

TABLE 1.3 – Caractérisation des réseaux

Le PAN est utilisé chez un particulier, le LAN dans un bâtiment (ou plusieurs bâtiments proches) d'une entreprise, le MAN interconnecte différents sites à l'échelle d'une agglomération, et le WAN s'étend sur un pays.

1.5 Topologies de réseaux

1.5.1 Composants

Les composants des réseaux se répartissent selon deux types :
- les *composants de traitement* sont les entités produisant et/ou consommant les informations qui circulent sur le réseau (par exemple les ordinateurs) ;
- les *composants de routage* assurent la transition et la circulation des informations échangées entre les composants de traitement (par exemple, les câbles, commutateurs).

1.5.2 Connexions

La connexion entre entités peut être *point-à-point*, c'est-à-dire qu'elle peut associer exactement deux entités, ou peut être une *connexion multipoints* qui en associe plus. Les *modes de communication* sont *simplex*, c'est-à-dire dans un seul sens, ou *duplex*, dans les deux sens.

1.5.3 Architectures des réseaux

L'*architecture d'un réseau* comprend 3 parties :

L'architecture physique définit la topologie physique d'*interconnexion* des composants du réseau.

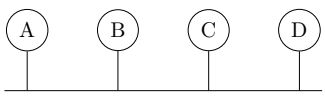
L'architecture logique définit la topologie de *circulation de l'information*. Elle peut être différente de l'architecture physique.

L'architecture logicielle définit les *logiciels assurant l'acheminement* des données.

Les architectures physiques et logiques les plus classiques sont le bus, l'étoile, l'anneau, l'arbre, le graphe et la topologie complète.

1.5.4 Le bus

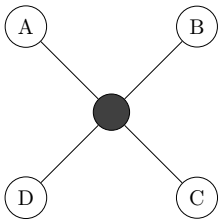
La topologie en bus consiste en un câblage unique auquel les différents nœuds sont connectés.



Le câble est l'unique élément matériel constituant le réseau et seuls les nœuds génèrent des signaux. Lorsqu'une station est en panne et ne transmet plus sur le réseau, elle ne perturbe pas le réseau. Par contre une seule coupure du câble empêche toute station d'échanger des messages sur le réseau.

1.5.5 L'étoile

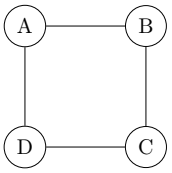
Dans la topologie en étoile tout nœud est connecté à un point central. Le point central doit être un matériel actif, comme un concentrateur ou un commutateur, c'est-à-dire un matériel qui remet en forme les signaux et les régénère avant de les retransmettre.



La panne d'un nœud ne perturbe pas le fonctionnement global du réseau. En revanche, une panne de l'équipement central qui relie toutes les stations rend le réseau totalement inutilisable.

1.5.6 L'anneau

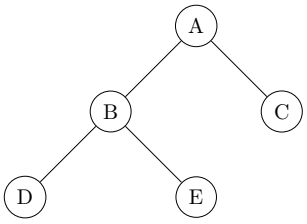
La topologie en anneau repose sur une boucle fermée, en anneau, où toutes les stations sont connectées en chaîne les unes aux autres par une liaison point-à-point.



Chaque station joue le rôle de nœud intermédiaire. Les informations transitent par chaque nœud, qui se comporte comme un répéteur et retransmet les informations au nœud suivant. La défaillance d'une station rend le réseau inutilisable.

1.5.7 L'arbre

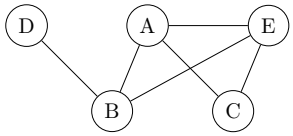
Dans l'architecture en arbre, les nœuds sont reliés entre eux de manière hiérarchique, donc par niveaux. Le sommet de haut niveau est appelé *racine*. Chaque nœud peut être connecté à plusieurs nœuds de niveau inférieur : dans ce cas on appelle ce nœud *parent* et les nœuds de niveau inférieur *fils*.



Le point faible de cette topologie est le fait que si le nœud parent tombe en panne, il paralyse tout le sous-arbre dont il est racine.

1.5.8 Le graphe

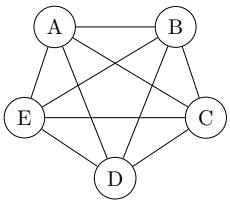
Dans la topologie en graphe les nœuds sont connectés entre eux par des liaisons point-à-point, de manière à constituer un graphe connexe.



L'information peut parcourir le réseau suivant des itinéraires différents.

1.5.9 La topologie complète

Dans la topologie complète il y a une liaison point-à-point entre chaque paire de nœuds, donc toute station est reliée à toutes les autres.



L'inconvénient de cette topologie est le nombre de liaisons nécessaires qui devient très élevé lorsque le nombre de stations augmente.

Chapitre 2

Modèle de référence OSI

2.1 Normalisation

2.1.1 Qu'est-ce et pourquoi ?

L'établissement de *normes* permet d'avoir une structure homogène pour faire communiquer différents équipements. La *conformité* à une norme garantit la satisfaction de règles précises.

Ainsi, des matériels différents, fabriqués par diverses entreprises, peuvent communiquer car la norme offre un cadre compatible entre ces entités hétérogènes.

La norme permet également d'assurer un niveau minimum de qualité.

2.1.2 Organismes de normalisation

La normalisation est effectuée par des organismes compétents au sein desquels les différents acteurs du domaine sont représentés. Trois organismes internationaux sont concernés par la normalisation dans le domaine des réseaux :

UIT (Union Internationale des Télécommunications) est l'institution spécialisée de l'ONU (Organisation des Nations Unies) dans le domaine des télécommunications. Elle comprend deux branches : l'UIT-T chargée de la normalisation dans le domaine des télécommunications et l'UIT-R qui s'occupe du domaine des radiocommunications.

IEC (International Electrotechnic Commission), fondée en 1906, est chargée de coordonner et d'unifier les normes dans le domaine de l'électricité.

ISO (International Standards Organisation) est une organisation privée chargée de la normalisation dans tous les domaines sauf l'électricité et l'électronique.

Ces organismes regroupent des représentants d'organismes nationaux. En France, les normes sont gérées par l'AFNOR (Association Française de NORmalisation).

2.2 Modèle de référence OSI

Le modèle de référence défini par l'ISO est l'OSI (*Open System Interconnection*). Il permet à des systèmes hétérogènes de s'interconnecter et d'échanger des informations. Il est par conséquent indépendant de la structure et de la technologie des matériels employés. Ce modèle offre un cadre permettant d'assurer une compatibilité maximum entre les entités communicantes tout en minimisant les contraintes permettant de les réaliser.

2.2.1 Principes de la structuration en couches

La complexité de conception, de réalisation et de maintenance des logiciels et de l'architecture des réseaux, est maîtrisée grâce à une organisation en *couches* ou *niveaux*, chaque couche étant

bâtie au-dessus de la précédente.

Le rôle de la couche de niveau N est de fournir des *services* à la couche de niveau $N+1$ tout en lui dissimulant les détails d'implémentation. La couche de niveau $N+1$ communique à la couche N les caractéristiques du service attendu.

La couche N, pour offrir les services à la couche $N+1$, utilise les services offerts par la couche $N-1$. Par conséquent, chaque couche peut *interagir uniquement avec les deux couches adjacentes.*

Une couche N est constituée d'un ensemble d'entités formant un sous-système de niveau N. Elle ne peut *dialoguer qu'avec une couche de même niveau* N sur une autre machine. Les communications se font donc entre *entités homologues.* La communication entre deux entités homologues de niveau N obéit à un ensemble de *règles et formats, syntaxiques et sémantiques*, prédéfinis pour les entités de niveau N. Ces règles et formats définissent le *protocole* de niveau N.

2.2.2 Couches du modèle OSI

Le modèle OSI est composé de *sept couches*, qui sont organisées de la façon suivante :

Niveau	Couche
7	Application
6	Présentation
5	Session
4	Transport
3	Réseau
2	Liaison de données
1	Physique

Chaque couche a un rôle spécifique :

7. La *couche application* offre aux utilisateurs des services normalisés pour la conception de leurs applications.

6. La *couche présentation* réalise la compression et le chiffrement, et vérifie la syntaxe des données échangées.

5. La *couche session* contrôle le dialogue entre les machines qui communiquent. Elle gère en particulier la synchronisation du dialogue et la reprise après interruption.

4. La *couche transport* assure le transport de bout en bout, c'est-à-dire entre les deux stations qui communiquent. Elle garantit que le message est acheminé entre les deux stations avec *la qualité de service* demandée. Le terme qualité de service désigne un ensemble de propriétés que le demandeur du service exige du prestataire, telles que la garantie d'un débit minimum, le respect d'une borne maximum de temps de livraison de messages, ...

3. La *couche réseau* assure l'acheminement des blocs d'information à travers le sous-réseau. Elle choisit le meilleur chemin entre les deux commutateurs d'entrée-sortie du sous-réseau. Les blocs d'information de niveau 3 sont appelés *paquets.*

2. La *couche liaison de données* est responsable de l'acheminement sans erreur des blocs d'information entre les deux machines qui se trouvent aux extrémités d'une liaison de données. Les blocs d'information de niveau 2 sont appelés *trames.*

1. La *couche physique* définit les moyens mécaniques (connecteurs), électriques et fonctionnels nécessaires à l'activation, au maintien et à la désactivation des connexions physiques destinées à la transmission des données binaires au niveau de la couche liaison de données. Elle fournit donc tous les éléments matériels et logiciels nécessaires au transport correct des données binaires, comme :
- les interfaces de connexion des équipements informatiques au support de transmission, appelées *jonctions* ;

— les supports de transmission ;
— les cartes réseaux ;
— les modems ;
— les multiplexeurs, qui concentrent plusieurs communications sur une ligne de transmission unique.

2.3 Interactions entre couches

2.3.1 Protocoles et services

D'un point de vue *logique* la communication intervient entre deux entités homologues, qui discutent en suivant des protocoles. Cela est illustré sur la figure 2.1.

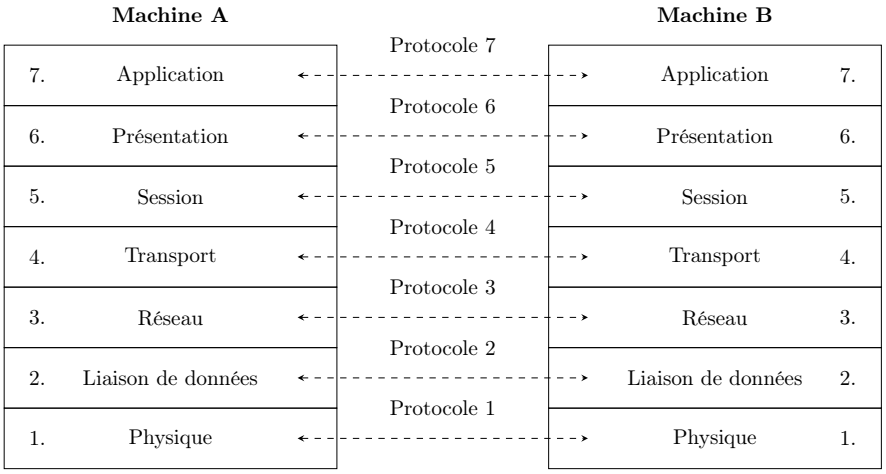
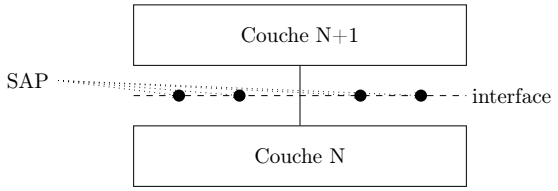


FIGURE 2.1 – Protocoles de communication entre couches de même niveau

En réalité aucune donnée ne passe directement de la couche N de la machine A à la couche N de la machine B. Chaque couche utilise les services de la couche immédiatement au-dessous et offre des services à la couche au-dessus. Les notions de protocole et de service sont donc fondamentales pour le fonctionnement de la pile des protocoles.

Définition 8

— *Un* protocole *de niveau N est un ensemble de* règles et formats, syntaxiques et sémantiques *prédéfinis qui sont utilisés dans la communication entre entités d'un même niveau N de deux machines différentes.*
— *Un* service *est fourni par une couche de niveau N à la couche de niveau $N+1$ d'une même machine.*
— *Les services fournis par une couche N sont identifiés par des* SAP (Service Access Point).
— *L'*interface *définit les opérations fondamentales et les services que la couche inférieure offre à la couche supérieure.*



La figure 2.2 décrit la communication entre les 7 niveaux de couches de deux entités communicantes A et B.

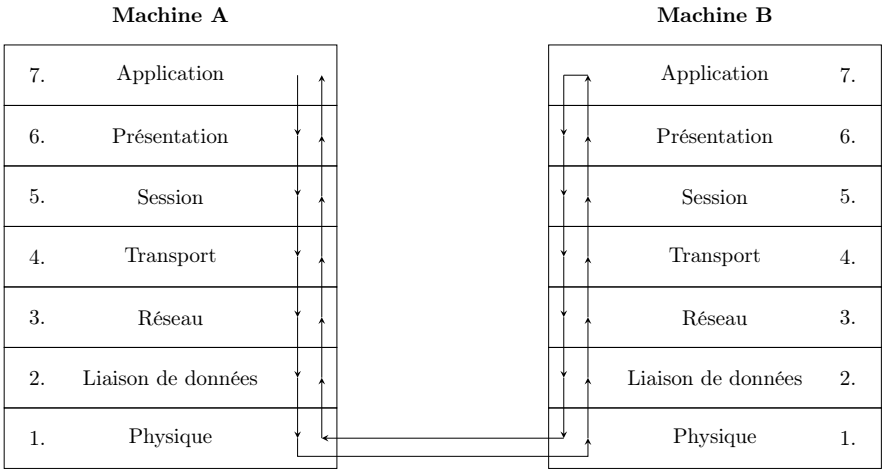


FIGURE 2.2 – Communication entre couches

2.3.2 Encapsulation, PDU et SDU

Les messages échangés par un protocole de niveau N sont appelés des PDU_N *(Protocol Data Unit de niveau N).*

Les messages échangés entre la couche N et la couche inférieure $N-1$ sont appelés des SDU_{N-1} *(Service Data Unit de niveau* $N-1$*).*

De plus, un protocole de niveau N ajoute au SDU_N qu'il a reçu des *informations de contrôle* visant à contrôler la bonne exécution du protocole. Ces informations de contrôle sont appelées PCI_N *(Protocol Control Information de niveau N).*

On a par conséquent :

$$PDU_N = SDU_N + PCI_N$$

$$SDU_N = PDU_{N+1}$$

On dit alors que le PDU_N *encapsule* le SDU_N.

Au lieu d'indexer le PDU ou le SDU par le numéro de la couche, on le fait souvent précéder de la première lettre du nom de la couche (en anglais). Par exemple, NPDU = PDU_3, où le N indique la couche réseau (*network*).

Exemple 1 *Considérons deux machines, A et B, qui communiquent en utilisant un modèle à 5 couches. Une application de niveau 5 de la machine A doit envoyer des données à sa couche homologue de la machine B. Supposons que toutes les couches rajoutent une en-tête (dénotée H comme* header*) et que les couches 1 et 2 rajoutent aussi une en-queue (dénotée T comme* tail*).*

La structure d'un message de niveau 5 sera donc la suivante :

H1	*H2*	*H3*	*H4*	*H5*	*données*	*T2*	*T1*

2.3.3 Primitives de service

Les protocoles peuvent opérer en *mode connecté* ou en *mode déconnecté*.

En mode connecté, la communication entre entités homologues de même niveau passe par l'établissement d'une *connexion*. Le transfert des données comporte donc trois phases :

1. l'établissement de la connexion,
2. le transfert des données,
3. la déconnexion.

Le contexte de la communication est préservé.

En mode déconnecté, seule la phase de transfert a lieu, et la communication s'effectue sans mémoire.

Primitives de service

Il existe 4 primitives de service : *requête* (en anglais : *request*), *indication* (*indication*), *réponse* (*response*) et *confirmation* (*confirmation*). Les protocoles qui fournissent des services avec confirmation utilisent les quatre primitives, en revanche les protocoles qui fournissent des services sans confirmation utilisent seulement les primitives « requête » et « indication ».

Une *requête* est initialement envoyée par la couche N à la couche $N-1$ d'une même entité. Ensuite, une *indication* est transmise de la couche $N-1$ à la couche N de l'autre entité communicant.

Ensuite, si le service est avec confirmation, la *réponse* est envoyée par la couche N à la couche $N-1$ de cette seconde entité. Enfin, une *confirmation* est transmise de la couche $N-1$ à la couche N de l'entité ayant émis la requête. Ceci est illustré sur la figure 2.3.

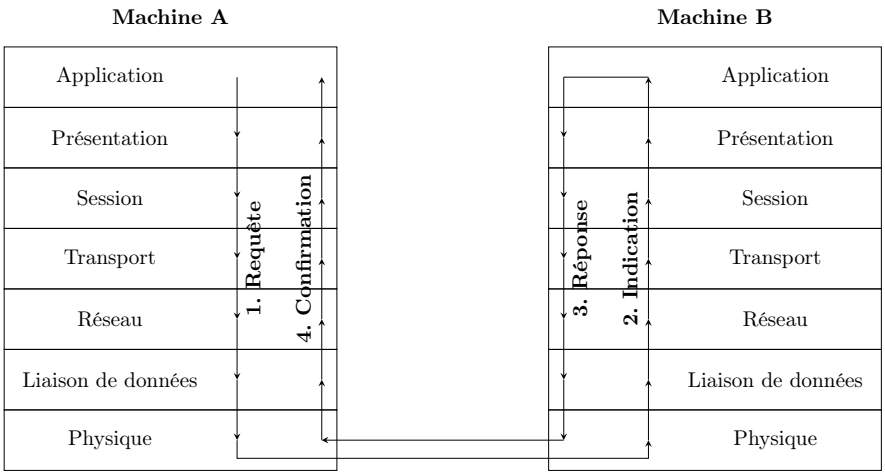


FIGURE 2.3 – Primitives de service

Syntaxe

Les commandes pour utiliser les services fournis par les couches suivent une syntaxe précise qui est définie dans la figure 2.4.

Format syntaxique	
Couche.Action.Primitive(arguments)	
où les arguments sont optionnels.	
Action	
Service avec connexion :	
CONNECT	établir une connexion
DATA(données)	transférer les données
DISCONNECT	libérer la connexion
Service sans connexion :	
UNIDATA(données)	pour transférer les données
Primitives de service	
Service confirmé :	
request	une entité sollicite un service
indication	une entité est informée d'un événement
response	une entité répond à un événement
confirm	une entité a bien reçu la réponse à sa demande
Service sans confirmation :	
request	une entité sollicite un service
indication	une entité est informée d'un événement

FIGURE 2.4 – Règles syntaxiques pour l'utilisation des services. **Remarque :** L'action CONNECT est toujours confirmée.

Chapitre 3

Modèle TCP/IP

Le modèle TCP/IP est nommé d'après ses deux protocoles principaux TCP et IP, mais il comporte en réalité plusieurs dizaines de protocoles. Il définit un modèle de *quatre couches*.

Niveau	Couche
4	Application
3	Transport
2	Internet
1	Hôte-réseau

Ce modèle a été proposé dix ans avant le modèle OSI et c'est le modèle couramment utilisé. La figure 3.1 montre comment le modèle TCP/IP se situe par rapport au modèle OSI.

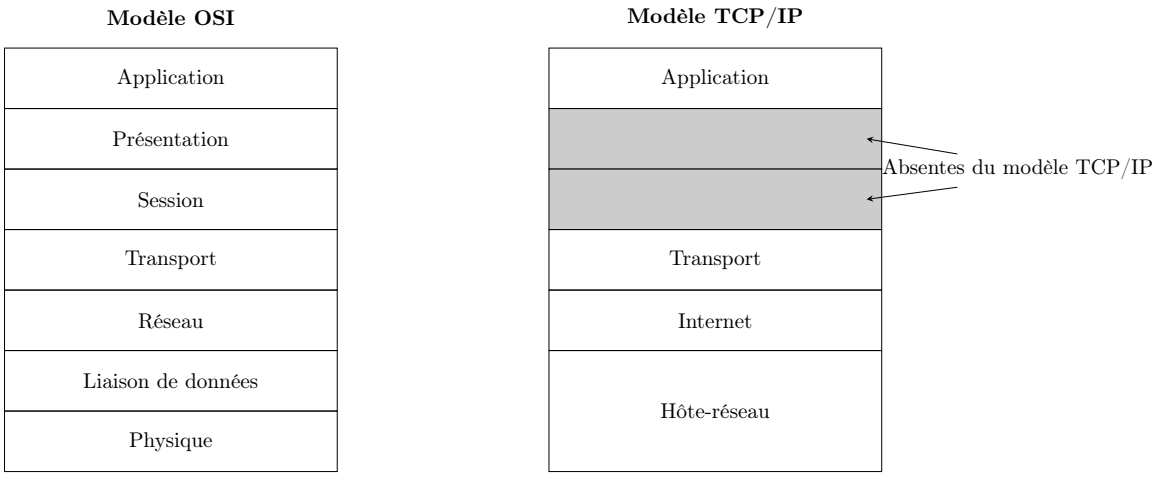


FIGURE 3.1 – Comparaison entre le modèle TCP/IP et le modèle OSI

3.1 Couche Hôte-réseau

La couche Hôte-réseau utilise un système d'adressage basé sur les adresses MAC.

Définition 9 *Une* adresse MAC *(Media Access Control) est un identifiant physique codé sur 6 octets qui est enregistré dans la carte réseau.*

Les adresses MAC sont souvent écrites en hexadécimal par 6 mots de 2 lettres séparés par un ":". Cette adresse identifie de façon unique l'interface réseau de la machine : les premiers trois octets constituent l'identifiant du constructeur de la carte, les derniers trois l'identifiant de la carte attribué par le constructeur.

Exemple 2 *L'adresse MAC 00000000 00011000 11011110 00010000 11111010 10000111 sera écrite* `00:18:DE:10:FA:87`*. L'identifiant du constructeur est* `00:18:DE` *(Intel), l'identifiant de la carte attribué par Intel est* `10:FA:87`*.*

Une machine avec plusieurs cartes réseaux aura plusieurs adresses MAC. Hors cas de falsification ou piratage, une adresse MAC est *unique* sur toute la planète.

L'adresse MAC de diffusion est `FF:FF:FF:FF:FF:FF`.

3.1.1 Ethernet

Le protocole Ethernet, normalisé comme IEEE 802.3, a fini par devenir omniprésent en entreprise. Il repose sur une topologie physique et logique en bus : il y a donc plusieurs machines connectées au même câble. Lorsqu'une machine A envoie une trame à une machine B toutes les stations du réseau vont la recevoir, mais seulement la machine B la traite (les autres l'ignorent).

Pour régler l'accès au canal physique, Ethernet utilise une technologie appelée *Carrier Sense Multiple Access with Collision Detection* (CSMA/CD).

Carrier Sense Multiple Access. Quand une station doit envoyer des données, elle exécute l'algorithme CSMA :

1. elle commence par écouter le support de transmission pour savoir si une autre station n'est pas déjà en train de transmettre.
2. Si c'est le cas, elle attend que le support se libère.
3. Quand le support est de nouveau disponible elle transmet sa trame.

Collision Detection. Néanmoins, si deux stations, après s'être assurées que le canal est libre, commencent une transmission au même moment elles provoqueront une *collision* qu'elles sauront toutefois détecter en écoutant le support. Lorsqu'une collision se produit, la station émettrice d'une trame lance la procédure de gestion des collisions :

1. elle continue la transmission à hauteur pour s'assurer que toutes les stations détectent la collision,
2. elle observe un temps de pause aléatoire dépendant du nombre de tentatives de transmission,
3. elle redémarre l'envoi de la trame.

Trame. La couche Ethernet ajoute à son SDU 22 octets d'en-tête et 4 octets d'en-queue, donc un total de 26 octets de PCI. Si le SDU encapsulé a une taille inférieure à 46 octets, alors des 0 sont rajoutés à la fin de telle manière à atteindre cette taille minimale. Ces octets à 0000 0000 sont appelés *octets de bourrage*. La trame Ethernet a donc la structure suivante :

Préambule	SFD	DA	SA	DL/EType	Données	Bourrage	FCS
7 octets	1 o.	6 o.	6 o.	2 o.	0–N o.	0-46 o.	4 o.
				en-tête	*SDU*		en-queue

où :

— Préambule : séquence de 7 octets à 1010 1010. Il permet la synchronisation des horloges de l'émetteur et du récepteur.
— SFD (*Start Frame Delimiter*) : un octet à 1010 1011 indiquant le début de la trame.
— DA : adresse MAC destination.
— SA : adresse MAC source.
— DL/Etype : si plus petit que 1 500 c'est la longueur des données (Ethernet I), sinon c'est le type de la trame (Ethernet II). Par exemple 0800 correspond à l'encapsulation d'un paquet IP.
— FCS (*Frame Check Sequence*) : utilisé par le code polynomial, voir section 4.4.2.

Remarque 2 *Le préambule et le SFD sont souvent considérés juste comme une phase de synchronisation et donc extérieurs à la structure de la trame.*

3.1.2 Token Ring

Le protocole *Token Ring* (anneau à jeton) a été normalisé comme IEEE 802.5. Il existe différents standards autorisant des débits de 4 Mbit/s, 16 Mbit/s, puis 100 Mbit/s qui sont fonction du câblage utilisé. Son concurrent Ethernet, moins onéreux et plus facile d'exploitation, quoique moins efficace, a fini par le supplanter, même si l'on retrouve encore Token Ring dans des infrastructures importantes.

Token Ring repose sur une topologie logique en anneau (voir section 1.5.6), où chaque machine se comporte comme un répéteur. Les trames parcourent l'anneau dans un sens qui est toujours le même. Pour régler l'accès au support physique, ce protocole utilise une trame spéciale de trois octets, appelée *jeton*, qui est répétée de poste en poste et circule dans le sens l'anneau.

Le fonctionnement du protocole est le suivant :

1. À la réception du jeton, une machine qui désire envoyer une trame conserve le jeton et envoie ses données.
2. Les données sont répétées de station en station jusqu'à atteindre son destinataire.
3. La trame est alors marquée pour préciser à l'émetteur si les données ont été reçues correctement.
4. La trame revient alors vers son émetteur qui constate que ses données sont effectivement arrivées à destination. L'émetteur peut éventuellement envoyer plusieurs trames, mais il ne peut garder le jeton que pendant une durée maximale.
5. Le jeton est remis en circulation et le processus peut recommencer.

La machine de l'anneau ayant l'adresse MAC la plus élevée joue le rôle de surveillant du jeton. Elle est en charge de résoudre un certain nombre de problèmes comme :
— la détection de la perte du jeton,
— la suppression d'une trame qui a déjà effectué un tour complet (l'émetteur a disparu avant de supprimer sa trame de l'anneau).

3.2 Couche Internet

3.2.1 Internet Protocol (IP)

Internet Protocol (IP) fournit un système de livraison de paquets, sans connexion et non fiable.

« Sans connexion » signifie qu'IP ne maintient aucune information d'état concernant les trames successives. Quand une machine envoie plusieurs trames à une autre machine, chaque trame est gérée indépendamment des autres, donc les trames peuvent emprunter des chemins différents, et l'ordre de réception peut être différent de l'ordre d'envoi.

« Non fiable » signifie qu'il n'existe aucune garantie que la trame IP arrive à destination avec succès. Toute fiabilité requise doit être assurée par les couches supérieures (par exemple en utilisant TCP).

Adressage. Le protocole IP gère des adresses logiques qui sont appelées *adresses IP*.

Définition 10 *Une adresse IP*[1] *est un numéro d'identification de quatre octets (IPv4) qui est attribué de façon permanente ou provisoire à chaque carte réseau utilisant le protocole IP.*

L'adresse IP est assignée soit manuellement par l'administrateur du réseau local, soit automatiquement via le protocole DHCP. Si un composant dispose de plusieurs interfaces, chacune dispose d'une adresse IP spécifique.

Remarque 3 *Les adresses IP sont généralement écrites sous forme* décimale pointée *: ce sont quatre entiers de 0 à 255 séparés par des points.*

Exemple 3 *L'adresse binaire 00001010 00000000 00000000 00000001 est notée 10.0.0.1*

La suite d'octets d'une adresse IP est divisée en deux parties :
— NET-ID : une sous-séquence qui désigne l'adresse d'un réseau.
— HOST-ID : une sous-séquence qui désigne l'adresse d'une machine sur le réseau désigné par le NET-ID.

Les adresses IP sont réparties en cinq classes. Les champs NET-ID et HOST-ID ont des longueurs variables, qui dépendent de la classe de l'adresse IP.

Classe A (grands réseaux) : ce sont les adresses dont le bit de poids fort est à 0. Le NET-ID est alors codé sur le premier octet et les trois derniers octets représentent le HOST-ID.
— Plage : de 1.0.0.0 à 127.255.255.255.

Classe B (réseaux moyens) : ce sont les adresses dont les deux bits de poids fort sont 10. Le NET-ID est alors codé sur 2 octets et le HOST-ID sur les deux autres octets.
— Plage : de 128.0.0.0 à 191.255.255.255.

Classe C (Petits réseaux) : ce sont les adresses dont les trois bits de poids forts sont 110. Le NET-ID est alors codé sur les 3 premiers octets et le HOST-ID sur le dernier octet.
— Plage : de 192.0.0.0 à 223.255.255.255.

Classe D (Multicast) : Ce sont les adresses dont les quatre bits de poids forts sont 1110. Les 28 bits qui restent désignent dans ce cas un groupe de multicast.
— Plage : de 224.0.0.0 à 239.255.255.255.

Classe E (Classe réservée) : ce sont les adresses dont les cinq bits de poids fort sont 1111. C'est une classe réservée pour des usages futurs.
— Plage : de 240.0.0.0 à 255.255.255.255.

Certaines adresses IP sont réservées pour des usages particuliers :
— 255.255.255.255 : adresse de diffusion.
— NET-ID $1 \cdots 1$: adresse de diffusion sur le réseau identifié par le NET-ID.
— NET-ID $0 \cdots 0$: adresse d'un réseau.
— 127.X.Y.Z : adresse de la boucle locale. Il permet de tester la pile de protocoles TCP/IP locale d'une machine sans passer par un réseau.
— 0.0.0.0 : valeur qui indique l'absence d'une adresse IP.

Définition 11 *Le* masque réseau *est une suite de quatre octets dont les bits correspondant à la partie NET-ID sont à 1, et ceux du HOST-ID sont à 0.*

En faisant un « et logique » bit à bit entre une adresse IP et le masque associé on obtient l'adresse du réseau auquel il appartient. La paire (adresse IP, masque) peut être représentée en notation CIDR (Classless Inter-Domain Routing) par l'adresse IP suivie d'une barre oblique « / » et le nombre de bits à 1 dans la notation binaire du masque de sous-réseau.

Exemple 4 *Le masque réseau associé à l'adresse 192.168.1.1 est 255.255.255.0 puisque c'est une adresse de la classe C. En notation CIDR on indiquera les deux comme 192.168.1.1/24. Un « et logique » entre 192.168.1.1 et 255.255.255.0 donne l'adresse réseau 192.168.1.0.*

1. Sauf mention contraire, on considère dans ce document les adresses IPv4.

Routage. Les réseaux locaux peuvent être reliés entre eux à travers des machines particulières que l'on appelle *routeurs*. Le protocole IP est capable de choisir la route suivant laquelle les paquets de données seront relayés de proche en proche jusqu'au destinataire. Donc le routage IP fonctionne de façon totalement décentralisée : aucun routeur n'a une vision globale de la route que prendront les paquets de données.

Chaque réseau a un certain *Maximum Transmission Unit* (MTU). Le MTU est la taille maximale d'un paquet pouvant circuler sur le réseau. Si la taille du paquet est plus grande que le MTU du réseau le protocole IP du routeur fragmente le paquet en plusieurs paquets avant de les injecter sur le réseau.

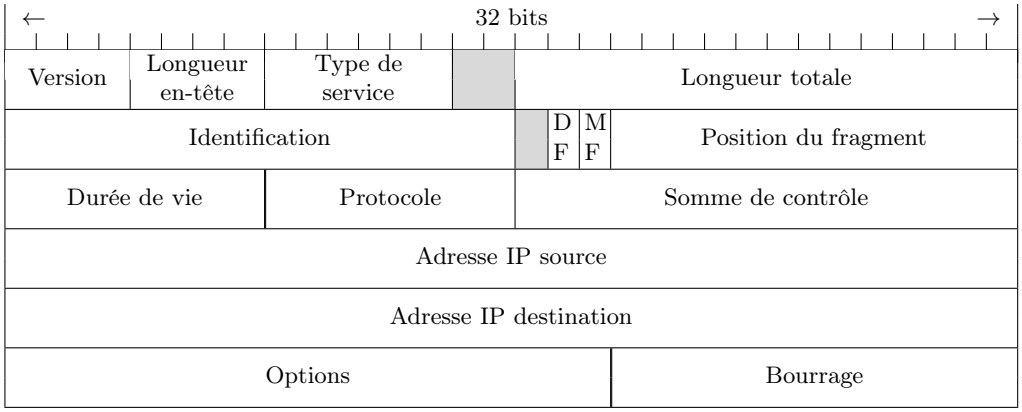
Une fois que tous les fragments sont arrivés à destination, le paquet original est reconstruit.

Les adresses IP des machines source et destinataire sont spécifiées dans l'en-tête IP des paquets. Chaque routeur possède une table interne, appelée *table de routage*, qui lui indique la ligne sur laquelle les paquets doivent être expédiés pour une destination donnée. La table contient des entrées qui associent l'adresse IP de destination et la ligne qu'il faut emprunter pour l'atteindre.

Dans le cas du *routage statique*, l'administrateur réseau doit initialiser la table de routage de chaque routeur en saisissant les entrées correctement. Ces routes ne changeront que si l'administrateur les modifie ensuite manuellement.

Dans le cas du *routage dynamique*, les routes sont calculées automatiquement par un algorithme qui est exécuté sur chaque routeur. Le protocole permet aux routeurs d'échanger des informations concernant l'état du réseau et utiliser ces informations afin de calculer la meilleure route possible et mettre à jour leurs tables de routage.

Paquet IP. Les paquets IP sont encapsulés dans Ethernet ou Token Ring. Le format d'un paquet IP est le suivant.



— Version : précise la version du protocole IP. La version couramment la plus utilisée est la 4.
— Longueur d'en-tête : donne la longueur de l'en-tête du paquet IP en mot de 4 octets.
— Type de service : informe les routeurs de la priorité du paquet et de la qualité de la route souhaitée.
— Longueur totale : donne la longueur (en octets) du paquet, en-tête comprise.
— Numéro d'identification : le couple (adresse IP source, identification) identifie un paquet de manière unique. Ces informations sont nécessaires pour effectuer l'assemblage des fragments d'un paquet qui a subi une fragmentation.
— DF (*Do not Fragment*) : si ce drapeau est positionné la fragmentation du paquet sera interdite.
— MF (*More Fragments*) : si ce drapeau est positionné, le paquet fait partie d'une série de fragments, mais ce n'est pas le dernier d'entre eux. Sinon, c'est le dernier fragment.
— Position du fragment (*Offset*) : indique la position du premier octet dans le paquet initial (non fragmenté), exprimée en multiple de 8 octets.
— TTL (*Time To Live*) : si la valeur de la durée de vie d'un paquet est N alors ce paquet peut encore traverser $N - 1$ routeurs.

— Protocole : indique la nature du protocole encapsulé dans le paquet IP. La liste des protocoles est donnée dans le RFC 1700. Quelques valeurs : ICMP (1), TCP (6) et UDP (17).
— Somme de contrôle : code de protection de l'en-tête IP.
— Adresse IP Source/Destination.
— Options : nous n'étudions pas ici ces options, mais il faut noter que ce champ peut être complété par des octets de bourrage afin de rendre la taille de l'en-tête IP un multiple de 4 octets puisque la longueur de l'en-tête est exprimée en mots de 4 octets.

3.2.2 Address Resolution Protocol (ARP)

Le protocole de résolution d'adresse (ARP, *Address Resolution Protocol*) effectue la traduction d'une adresse de protocole de couche réseau (typiquement une adresse IPv4) en une adresse MAC (typiquement une adresse Ethernet).

ARP : Adresse Logique → Adresse Physique

Ce protocole permet à une machine de trouver l'adresse physique d'une autre machine (connectée sur le même réseau) à partir de son adresse IP. Il se situe à l'interface entre la couche réseau (couche 3 du modèle OSI) et la couche de liaison (couche 2 du modèle OSI).

Le fonctionnement du protocole est le suivant :

1. La machine source demande l'adresse physique correspondant à une certaine adresse IP en envoyant en mode diffusion un paquet *ARP-Request*.
2. Le paquet ARP-Request contient l'adresse IP de la machine recherchée ainsi que les adresses IP et physique de la machine source.
3. Si la machine recherchée se trouve sur le réseau local alors elle répond directement à la machine source en envoyant un paquet *ARP-Reply* qui contient son adresse physique.

Un mécanisme de *cache* permet de réduire les délais d'attente dus à l'exécution du protocole. La machine garde en mémoire pour un certain temps les correspondances obtenues entre les adresses logiques et les adresses physiques.

Message ARP. Les messages ARP sont encapsulés directement dans Ethernet ou Token Ring. Le format du message ARP est le suivant.

Type Mat. 2 o.	Type Prot. 2 o.	Long. Adr. Phy. 1 o.	Long. Adr. Prot. 1 o.	OP 2 o.	Adr. Phy. Source 6 o.	Adr. Prot. Source 4 o.	Adr. Phy. Dest. 6 o.	Adr. Prot. Dest. 4 o.

où :
— Type Mat. : type de matériel physique (par exemple : Ethernet, Token Ring, etc.),
— Type Prot. : type de protocole (par exemple : IP),
— Long. Adr. Phy. : longueur de l'adresse physique,
— Long. Adr. Prot. : longueur de l'adresse du protocole,
— OP : opération,
— Adr. Phy. Source : adresse physique de l'émetteur,
— Adr. Prot. Source : adresse du protocole de l'émetteur,
— Adr. Phy. Dest. : adresse physique du destinataire,
— Adr. Prot. Dest. : adresse du protocole du destinataire.

Exemple 5 *Une trame ARP a la structure suivante :*

en-tête Ethernet	*message ARP*	*en-queue Ethernet*

Calculons sa taille, en considère aussi le préambule et le SFD.

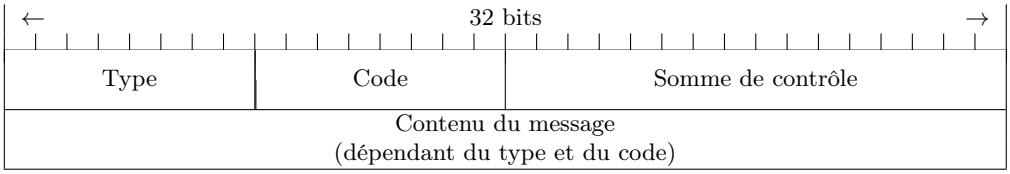
Le message ARP a une taille de $2+2+1+1+2+6+4+4 = 22$ *octets. L'en-tête et l'en-queue Ethernet comportent* $7+1+6+6+2+4 = 20$ *octets. De plus, le message ARP ayant une taille inférieure à* 46 *octets, il faut ajouter* $46-22 = 24$ *octets de bourrage. Donc au total, la trame ARP a une longueur de* $22+20+24 = 66$ *octets.*

Remarque 4 *Le protocole RARP (*Reverse Address Resolution Protocol*) effectue la traduction inverse, c'est-à-dire des adresses physiques en adresses logiques.*

3.2.3 Internet Control Message Protocol (ICMP)

Lorsqu'un événement inattendu se produit pendant la communication entre les entités d'un réseau, il est signalé par le protocole de messages de contrôle de l'Internet, ICMP (*Internet Control Message Protocol*). Plus généralement, ce protocole permet d'envoyer des messages de collecte d'informations sur l'état du réseau.

Message ICMP. Les messages ICMP sont encapsulés dans IP. Un message ICMP a le format suivant.



Le champ *type* spécifie le type de message ICMP. Certains types de messages ICMP se servent du champ *code* pour préciser la nature de l'erreur signalée. Le champ *somme de contrôle* est calculé sur l'intégralité du message ICMP et est utilisé pour vérifier s'il y a eu des erreurs de transmission du message. Le contenu du message peut varier selon le type et le code de l'erreur rencontrée.

Le tableau suivant montre les principaux types de messages ICMP.

Type de Message	**Description**
Destination inaccessible	Le paquet n'as pas pu être livré
Délai expiré	Le paquet a expiré
Demande d'écho	Demande à une machine si elle est active
Envoi d'écho	La machine distante est active

3.3 Couche Transport

Au niveau transport, il y a deux principaux protocoles : TCP (en mode connecté) et UDP (en mode non connecté). Les SAP de la couche transport sont des identifiants sur 16 bits appelés *ports*. Un port désigne l'application sur la machine qui a fait appel au protocole de transport. Les numéros de port de 0 à 1023 sont réservés pour les services systèmes (`http` sur le port 80, `telnet` sur le port 23, etc.)

Une connexion entre deux applications est caractérisée de manière unique par :
— les adresses IP source et destination,
— les numéros de ports source et destination,
— le protocole de transport utilisé : UDP ou TCP.

3.3.1 Transmission Control Protocol (TCP)

Transmission Control Protocol (TCP) est un protocole de transport qui assure un service orienté connexion avec confirmation. Il garantit la livraison sans erreur à n'importe quel hôte du réseau d'un flot d'octets émis par une machine. Il fragmente le flot d'octets entrant en messages

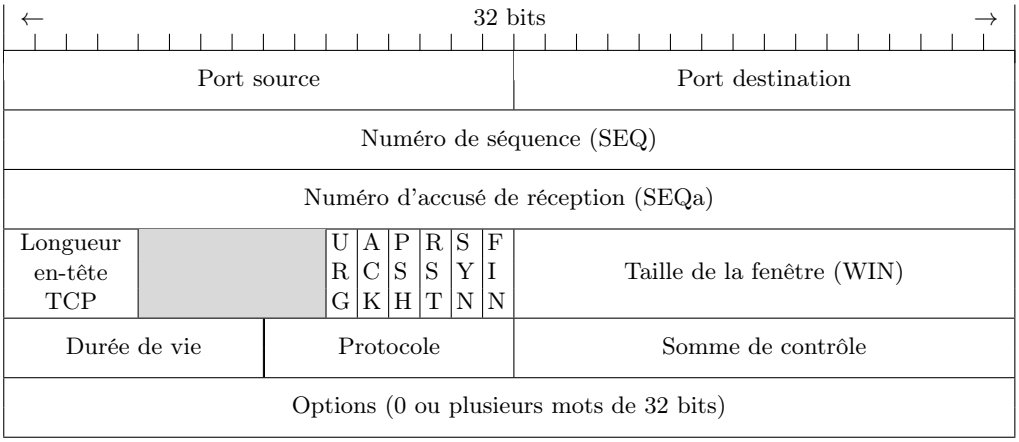
qu'il passe à la couche Internet. À l'arrivée, le processus TCP destinataire réassemble les messages reçus en un flot de sortie. TCP assure aussi un contrôle de flux pour éviter qu'un émetteur rapide ne submerge un récepteur lent de plus de paquets que celui-ci ne peut en traiter.

Établissement de la connexion. Tout d'abord, une connexion est ouverte entre le client et le serveur.

Transfert des données. Une fois que la connexion est établie, le protocole procède à l'envoi des messages. Pour accélérer la transmission et faire en sorte que l'émetteur n'attende pas de recevoir l'accusé de réception pour envoyer le message suivant, une notion de *fenêtre glissante* est utilisée. Le protocole TCP permet donc des *acquittements groupés*.

Déconnexion. Après communication, la connexion est terminée normalement par un dialogue entre le client et son serveur, avec une annonce de fin d'envoi, et l'accusé de réception correspondant.

En-tête TCP. Les messages TCP sont encapsulées dans IP. Le format TCP est le suivant.



— Port source : numéro de port source.
— Port destination : numéro de port destination.
— Numéro de séquence : indique le numéro du premier octet transmis dans le message. À l'ouverture de connexion un numéro de séquence initial (ISN, *Initial Sequence Number*) est attribué d'une manière aléatoire.
— Numéro d'accusé de réception : contient le numéro de la séquence du prochain octet attendu.
— Longueur de l'en-tête : indique la longueur de l'en-tête TCP en mot de 4 octets.
— Six bits de drapeaux :
  — URG (*Urgent*) : si positionné, indique que la valeur du champ message urgent est significative,
  — ACK (*Acknowledgement*) : si positionné, indique que la valeur du champ acquittement est significative,
  — PSH (*Push*) : Les données reçues doivent être transmises immédiatement à la couche supérieure,
  — RST (*Reset*) : fermeture de la connexion suite à une erreur,
  — SYN (*Synchronise*) : ouverture de la connexion,
  — FIN (*Finish*) : fin de la connexion.
— Fenêtre : indique le nombre d'octets que le récepteur peut accepter. Ce champ permet de gérer le contrôle de flux. Ainsi la valeur de la fenêtre indique à l'émetteur le nombre d'octets qu'il peut envoyer sans que ceux-ci soient acquittés.
— Pointeur message urgent : indique des octets qui doivent être traités en priorité.

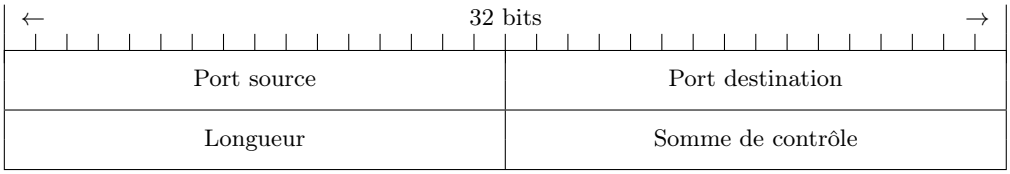
Exemple 6 *Un message TCP a la structure suivante :*

en-tête Ethernet	*en-tête IP*	*en-tête TCP*	*données TCP*	*en-queue Ethernet*

3.3.2 User Datagram Protocol (UDP)

Le protocole UDP (*User Datagram Protocol*) est un protocole de transport non orienté connexion ; il opère sans confirmation. Il y a donc seulement la phase d'envoi des données. Cela permet de gagner en débit pour les transmissions gourmandes, telles que vidéos et sons.

En-tête UDP. Les messages UDP sont encapsulées dans IP. Le format UDP est le suivant.



— Port source,
— Port destination,
— Longueur : indique la longueur totale du message (données et en-tête compris).
— Somme de contrôle : FCS pour protéger le message UDP.

3.4 Couche Application

Les protocoles de la couche Application seront traités dans d'autres cours. Une liste des principaux protocoles de cette couche est néanmoins donnée ci-dessous :

SMTP (*Simple Mail Transfer Protocol*) est un protocole de transfert simple, utilisé par la messagerie électronique (e-mail). Il repose sur TCP et IP.

POP3 (*Post Office Protocol 3*) est un protocole qui permet de récupérer les courriers électroniques situés sur un serveur de messagerie électronique.

IMAP (*Internet Message Access Protocol*) permet aux messages électroniques d'être stockés et conservés sur le serveur de messagerie, plutôt que d'être transférés sur le poste client.

HTTP (*HyperText Transfer Protocol*) est un protocole de communication client-serveur développé pour le World Wide Web.

HTTPS (*Secure HyperText Transfer Protocol*) est la variante de HTTP sécurisée par l'utilisation des protocoles SSL ou TLS.

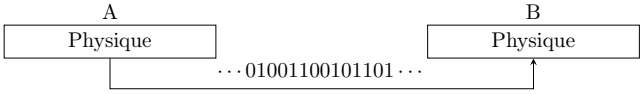
FTP (*File Transfer Protocol*) est un protocole de communication destiné à l'échange de fichiers sur le réseau.

Chapitre 4

Détection et Correction d'erreurs

4.1 La couche liaison de données

La couche 3 du modèle OSI (couche réseau) a pour mission, entre autres, de trouver le meilleur chemin pour acheminer les messages. Cette tâche est complexe et met en jeu plusieurs protocoles. La couche 3 d'un nœud A (ordinateur ou routeur) du réseau envoie le message à la couche 3 d'un autre nœud, B (ordinateur ou routeur), qui lui est physiquement connecté. Pour cela, elle doit passer le message à la couche physique dont le rôle est d'acheminer les bits.



Toutefois, le support matériel utilisé par la couche physique (câble coaxial, paire de cuivre torsadée, fibre optique, ondes radio, . . .) n'est pas fiable à 100%. Certains bits reçus peuvent être erronés (un 0 à la place d'un 1 ou inversement).

Le *service* rendu par la *couche liaison de données* est précisément de faire croire à la couche réseau qu'elle utilise une couche physique parfaite.

Pour cela, la couche liaison de A ajoute au SDU_2 transmis par la couche réseau un en-tête (*header*) et un en-queue (*trailer*). Ceux-ci sont interprétés par la couche liaison de B ; une fois que celle-ci est certaine qu'il n'y a pas d'erreurs, elle les supprime et passe les données à la couche réseau de B. Celle-ci, à son tour, cherche le meilleur chemin pour atteindre le destinataire final, et comprend qu'elle doit faire passer le message à un autre nœud (C) ; elle fait passer les données à sa couche 2, etc.

La couche liaison utilise principalement deux méthodes :

— la détection/correction d'erreurs, qui fait l'objet de ce chapitre ;

— un « dialogue » entre émetteur et récepteur, visant à s'assurer que la transmission se passe correctement : envoi d'accusé de réception, retransmission éventuelle de messages, etc.

4.2 Généralités sur les codes détecteurs/correcteurs d'erreurs

Un *code* détecteur/correcteur d'erreurs est un groupe de bits que l'émetteur ajoute au message à transmettre. Ce groupe de bits, appelé *FCS* (*Frame Check Sequence*), dépend du message et introduit de la *redondance*.

Définition 12 (Rendement d'un code) *Si le message comporte m bits et le FCS a une longueur de k bits, alors on dit que le* rendement *du code est $\frac{m}{m+k}$.*

Le FCS doit permettre de *détecter* des erreurs parmi les bits reçus, et éventuellement les localiser et les *corriger*. Mais s'il y a trop d'erreurs, la détection et la correction peuvent être impossibles.

4.2.1 Un code simple : la répétition

Une approche naïve consiste à *dupliquer* (c'est-à-dire répéter) le message à transmettre.

Supposons que le message effectivement transmis soit le double du message réel. Par exemple, pour envoyer 11100010, on transmet 1110001011100010. La détection et la localisation des erreurs sont alors simples : on cherche les différences entre les première et seconde moitiés du message. Par contre, il est impossible de corriger une erreur détectée : le bit erroné est différent dans les deux copies, et rien ne permet de dire lequel est le bon. Le rendement de ce code est $0,5$.

Pour remédier à ce problème, on peut envoyer le message en 3 exemplaires au lieu de 2. Dans ce cas, un bit a soit la même valeur dans toutes les copies, soit la même valeur dans deux d'entre elles et l'autre valeur dans la troisième copie. Le bit correct est probablement celui qui apparaît en deux exemplaires : on peut cette fois corriger l'erreur. Le rendement de ce code est $0,33$.

4.2.2 Inconvénients et problèmes rencontrés

Le code de répétition est simple, mais présente de nombreux inconvénients, illustrant ceux que l'on peut rencontrer avec les autres codes :

— le rendement est faible (respectivement $0,5$ et $0,33$) ;
— certaines erreurs peuvent ne pas être détectées ;
— certaines erreurs détectées ne peuvent pas être corrigées, voire être mal corrigées ;
— la correction nécessite plus de redondance que la détection d'erreurs.

4.3 Codes à contrôle de parité

Les codes à contrôle de parité sont soit *pairs*, soit *impairs*. Dans le premier cas, on protège une séquence de bits en ajoutant un nouveau bit, appelé *bit de parité*, de telle sorte que le nombre de bits ayant la valeur 1 dans la séquence protégée plus le bit introduit soit pair. Dans le second cas, ce nombre doit être impair.

4.3.1 VRC (*Vertical Redundancy Check*)

C'est la technique la plus simple. Un code ASCII étant défini sur 7 bits, on utilise le $8^{\text{ème}}$ bit de l'octet pour introduire le code vérificateur.

Exemple 7 *Pour transmettre la chaîne de caractères `IUT`, on code chaque lettre en ASCII, puis on ajoute le code de parité.*

Lettre	*ASCII*	*VRC pair*	*VRC impair*
I	*1001001*	**1***1001001*	**0***1001001*
U	*1010101*	**0***1010101*	**1***1010101*
T	*1010100*	**1***1010100*	**0***1010100*

Pour envoyer le message avec un code de parité pair, on transmet (avec l'ordre d'envoi des bits de gauche à droite) :

$$\underbrace{\mathbf{1}1001001}_{I}\ \underbrace{\mathbf{0}1010101}_{U}\ \underbrace{\mathbf{1}1010100}_{T}$$

Ce code permet de détecter les erreurs en nombre impair sans pouvoir corriger. Il est peu efficace.

4.3.2 LRC (*Longitudinal Redundancy Check*)

Le principe est similaire à celui du VRC, mais au lieu de protéger les caractères un par un, on protège l'ensemble des bits de même rang de tous les caractères. On obtient alors un code de protection sur 7 bits.

Exemple 8 *Pour protéger IUT, on calcule le code :*

I	*1001001*
U	*1010101*
T	*1010100*
LRC pair	*1001000*
LRC impair	*0110111*

Pour envoyer le message avec un code de parité pair, on transmet :

$$\underbrace{1001001}_{I}\ \underbrace{1010101}_{U}\ \underbrace{1010100}_{T}\ \underbrace{\mathbf{1001000}}_{LRC}$$

L'efficacité du code LRC dépend fortement de la longueur du message transmis.

4.3.3 LRC et VRC

On peut également combiner les deux techniques précédentes. On protège alors chaque caractère par un code VRC et l'ensemble des bits par un code LRC. On obtient donc un LRC sur 8 bits. La parité des LRC et VRC utilisés est la même (tous les deux pairs ou tous les deux impairs).

Exemple : Pour transmettre la chaîne de caractères `IUT`, on code chaque lettre en VRC puis en LRC :

		VRC pair	VRC impair
I	1001001	**1**1001001	**0**1001001
U	1010101	**0**1010101	**1**1010101
T	1010100	**1**1010100	**0**1010100
LRC		01001000	00110111

Pour envoyer le message avec un code de parité pair, on transmet :

$$\underbrace{\mathbf{1}1001001}_{I}\ \underbrace{\mathbf{0}1010101}_{U}\ \underbrace{\mathbf{1}1010100}_{T}\ \underbrace{\mathbf{01001000}}_{LRC}$$

4.4 Codes en blocs

4.4.1 Le code de Hamming

Quand on utilise un *code en blocs*, la suite de bits à émettre est divisée en messages (blocs) de m bits ; k *bits de contrôle* sont ajoutés à chaque message de m bits pour former un *mot du code* de $m + k$ bits. Chaque bit de contrôle est un bit de parité calculé sur plusieurs bits du message (mais pas tous les bits).

Prenons l'exemple d'un code découpant les bits à envoyer en blocs (ou messages) de 4 bits. Les messages possibles sont :
0000 - 0001 - 0010 - 0011 - 0100 - 0101 - 0110 - 0111 - 1000 - 1001 - 1010 - 1011 - 1100 - 1101 - 1110 - 1111.
Pour chacun d'entre eux, l'émetteur calcule des bits de contrôle, formant ainsi un mot du code. Considérons le code dont les mots sont donnés ci-dessous :

message	**bits de contrôle**	**mot du code**
0000	000	0000 000
0001	101	0001 101
0010	111	0010 111
0011	010	0011 010
0100	011	0100 011
0101	110	0101 110
0110	100	0110 100
0111	001	0111 001
1000	110	1000 110
1001	011	1001 110
1010	001	1010 001
1011	100	1011 100
1100	101	1100 101
1101	000	1101 000
1110	010	1110 010
1111	111	1111 111

Ce code génère un mot de 7 bits pour un message de 4 bit, il est donc appelé *code de Hamming(7,4)*. Le récepteur, à chaque fois qu'il reçoit un mot de 7 bits, vérifie que ce mot fait bien partie du code, c'est-à-dire de la troisième colonne du tableau ci-dessus. Si ce n'est pas le cas, il en déduit qu'un bit (au moins) est erroné. L'erreur est ainsi détectée. Comment le récepteur procède t-il pour corriger l'erreur ? Le code de Hamming, pour corriger les erreurs, se base sur une mesure de dissimilarité entre deux séquences de bits de même longueur, appelée *distance de Hamming*.

Définition 13 (distance de Hamming) *La* distance de Hamming *entre deux séquences binaires s_1 et s_2 de même taille est le nombre de bits de même rang pour lesquels ces deux séquences diffèrent. Elle est notée $d(s_1, s_2)$.*

Exemple 9 $d(1\mathbf{10}0110, 1\mathbf{01}0110) = 2$.

Le récepteur analyse donc le nombre de bits différents entre le code erroné reçu et chaque mot du code, et choisit un mot valide ayant la distance de Hamming minimale.

Exemple 10 *Imaginons par exemple que le mot envoyé soit* 1110010*. Le dernier bit est mal transmis et le récepteur reçoit* 1110011*. Il s'aperçoit que ce mot ne fait pas partie du code, puis il parcourt le tableau en calculant à chaque fois la distance entre le mot reçu et le mot du code. Il aboutit aux résultats suivants :*

mot du code	*distance avec 1110011*
0000 000	*5*
0001 101	*6*
0010 111	*3*
0011 010	*2*
0100 011	*2*
0101 110	*5*
0110 100	*4*
0111 001	*3*
1000 110	*4*
1001 110	*3*
1010 001	*2*
1011 100	*5*
1100 101	*3*
1101 000	*3*
1110 010	*1*
1111 111	*2*

Le récepteur en déduit que le message envoyé était 1110010*, car la distance de Hamming est 1, et donc corrige l'erreur.*

Que se serait-il passé si deux mots du code s'étaient trouvés à une distance de 1 du code 1110011 ? Comment corriger l'erreur dans ce cas ? Avec le code ci-dessus, cette situation ne se produit jamais, car chaque mot est séparé des autres par une distance de 3 au moins. On dit que la distance du code *est de 3.*

Définition 14 (distance d'un code) *Soit un code C comportant n séquences valides. La* distance de Hamming du code *C est la distance minimale séparant deux mots valides du code. Elle est notée $d(C)$.*

Notons enfin qu'il n'est pas toujours possible de corriger la ou les erreurs Par exemple, si le message envoyé est 1111111, mais deux erreurs le transforment en 1110011, le récepteur pensera à nouveau que le message était 1110010. Dans ce cas, la correction sera mauvaise. Le code de Hamming(7,4) est capable de corriger une erreur, mais pas deux. Plus généralement :

Propriété 1 *Un code de distance $d(C)$ détecte $d(C)-1$ erreurs et corrige $k = \lfloor \frac{d(C)-1}{2} \rfloor$ erreurs.*

Exemple 11 *Supposons que l'on souhaite transmettre des messages pouvant se coder comme l'une des séquences du code :*

$C = \{000000, 001110, 011011, 100011, 101101\}$.

$d(C) = \min_{(c_1,c_2)\in C^2} d(c_1, c_2) = 3$.

Avec ce code, on peut détecter au maximum $d(C)-1 = 2$ erreurs et en corriger $k = \frac{d(C)-1}{2} = 1$.

4.4.2 Codes polynomiaux

Les *codes polynomiaux* sont des cas particuliers de codes en blocs. Dans le code de Hamming(7,4), le calcul des bits de contrôle pourrait être fait de façon logicielle avec une structure conditionnelle du type `switch ... case` et le programme de codage en langage C serait très simple. Par contre si les blocs ne sont pas des groupes de 4 bits, mais de plusieurs centaines, voire plusieurs milliers de bits, le passage en revue des différents cas prend un temps considérable !

Les codes polynomiaux permettent de régler ce problème en utilisant un algorithme de calcul des bits de contrôle différent, et beaucoup plus rapide. Les codes polynomiaux possèdent un autre avantage : ils peuvent opérer sur des blocs de taille variable. C'est un cas qui se présente souvent en transmission, puisque la taille des données peut varier.

Un *code polynomial* est basé sur l'utilisation d'un *polynôme générateur* $G(x)$. Les polynômes manipulés sont binaires : tous les coefficients sont 0 ou 1. Par conséquent, un polynôme générateur de *degré* k s'écrit sous la forme :

$$G(x) = a_0 \oplus a_1.x \oplus a_2.x^2 \oplus a_3.x^3 \oplus \ldots \oplus a_k.x^k$$

où $$\forall i \in \{0..k\}, a_i \in \{0,1\}$$

Le polynôme $G(x)$ est associé à une valeur binaire.

Exemple 12 *La valeur binaire associée au polynôme* $G(x) = x^3 \oplus x \oplus 1$ *est* 1011.

Soit M le message (séquence de bits) à protéger. Un polynôme $M(x)$ lui est associé :

$$M = m_n \ldots m_2 m_1 m_0 \Rightarrow M(x) = m_n.x^n \oplus \cdots \oplus m_2.x^2 \oplus m_1.x \oplus m_0$$

Exemple 13 *Au message* $M = 1101$ *est associé le polynôme* $M(x) = x^3 \oplus x^2 \oplus 1$.

Codage

Le calcul du CRC s'effectue dans le corps $\mathbb{Z}/2\mathbb{Z}$, c'est-à-dire que :
- $1 \oplus 1 = 0$
- $x \oplus x = 0$
- $x = -x$

Soient :
— $G(x)$ un polynôme générateur de degré k ;
— $M(x)$ le polynôme associé au message M à transmettre.

La procédure de codage consiste à :
— calculer $P(x) = M(x).x^k$. Ceci correspond à un décalage de k bits (vers la gauche) du message M. La longueur du CRC (*Cyclic Redundancy Check*) calculé sera aussi de k bits. Cette opération de décalage revient à préparer la place nécessaire pour ces k bits de CRC.
— diviser le polynôme $P(x)$ par $G(x)$. Soient $Q(x)$ et $R(x)$ les polynômes quotient et reste ainsi obtenus :

$$P(x) = Q(x).G(x) \oplus R(x)$$

— le CRC est le reste $R(x)$ ainsi calculé. On remarque que le reste est forcément au maximum de degré $k-1$.
— le message effectivement transmis est associé au polynôme $M'(x) = P(x) \oplus R(x)$. Il est par conséquent composé du message initial M suivi de la séquence de k bits correspondant à $R(x)$.

Exemple 14 *Soient le polynôme générateur* $G(x) = x^3 \oplus x \oplus 1$ *et le message à envoyer* $M = 1101$. *Le polynôme correspondant au message est* $M(x) = x^3 \oplus x^2 \oplus 1$. *Le degré de* $G(x)$ *est* 3. *Donc,* $P(x) = M(x).x^3 = x^6 \oplus x^5 \oplus x^3$. *Effectuons la division de* $P(x)$ *par* $G(x)$ *:*

$$\begin{array}{rccccccc|l}
 & x^6 & \oplus x^5 & & \oplus x^3 & & & & x^3 \oplus x \oplus 1 \\
\oplus & x^6 & & \oplus x^4 & \oplus x^3 & & & & \hline x^3 \oplus x^2 \oplus x \oplus 1 \\
\hline
 & & x^5 & \oplus x^4 & & & & & \\
 & \oplus & x^5 & & \oplus x^3 & \oplus x^2 & & & \\
\hline
 & & & x^4 & \oplus x^3 & \oplus x^2 & & & \\
 & & \oplus & x^4 & & \oplus x^2 & \oplus x & & \\
\hline
 & & & & x^3 & & \oplus x & & \\
 & & \oplus & & x^3 & & \oplus x & \oplus 1 & \\
\hline
 & & & & & & & 1 &
\end{array}$$

Le quotient est donc $Q(x) = x^3 \oplus x^2 \oplus x \oplus 1$, *et le reste* $R(x) = 1$. *Le message transmis a alors pour polynôme* $M'(x) = x^6 \oplus x^5 \oplus x^3 \oplus 1$, *d'où* $M' = 1101001$.

Sur Ethernet, la taille des données à coder peut varier de 60 à un peu plus de 1500 octets, et le polynôme générateur utilisé est de degré 32. C'est un polynôme générateur un peu particulier, dit *cyclique*, d'où le nom du code correcteur d'erreur. Le calcul est effectué juste avant l'encodage physique des bits, par un circuit situé sur la carte réseau.

Décodage (ou détection)

Le récepteur pourrait exécuter un algorithme passant en revue tous les mots du code pour vérifier si le mot reçu est erroné ou non. Mais là encore, cette solution serait trop lente pour des blocs de taille importante. On préfère utiliser la propriété ci-dessous.

Propriété 2

1. *Un message M' transmis correctement a un polynôme $M'(x)$ divisible par le polynôme générateur $G(x)$.*
2. *Si $G(x)$ comporte au moins 2 termes, les erreurs simples sont détectables.*
3. *Si $G(x)$ a un facteur irréductible de 3 termes, les erreurs doubles sont détectables.*
4. *Si $G(x)$ est un multiple de $x \oplus 1$, les erreurs en nombre impair sont détectables.*

Lorsque l'on reçoit un message M', deux cas peuvent se présenter :
— $M'(x)$ est divisible par $G(x)$: le CRC ne permet pas de détecter une erreur. Il y a de fortes chances que le message reçu soit correct. Le message initial M est obtenu en ignorant les k derniers bits de M'.
— $M'(x)$ n'est pas divisible par $G(x)$: une erreur est détectée.

4.5 Choix d'un code et d'une stratégie de correction

Plus la distance d'un code est grande, plus le code peut détecter et corriger des erreurs, mais plus son rendement est faible. Le choix d'un code résulte donc d'un compromis entre ces deux paramètres, capacité à détecter/corriger et rendement.

Certains protocoles font le choix d'utiliser un code de faible rendement, mais de grande distance, capable de détecter et corriger des erreurs. Cette stratégie est dite FEC (*Forward Error Correction*). Elle est adoptée sur les supports de stockage type CD et DVD.

Une autre stratégie consiste à utiliser un code de bon rendement, mais de faible distance, par exemple une distance de 2. Un tel code peut détecter une seule erreur et est incapable d'effectuer des corrections. Si une erreur est détectée, le récepteur demande à l'émetteur de retransmettre le message (remarquons que cette procédure de correction par demande de retransmission ne peut pas se passer d'un code détecteur d'erreurs). Cette stratégie est dite ARQ (*Automatic Repeat Request*) et constitue un bon choix si les erreurs sont peu fréquentes. La stratégie ARQ est utilisée dans la plupart des réseaux informatiques (LAN et WAN).